

Using the Cray Programming Environment to Convert an all MPI code to a Hybrid-Multi-core Ready Application

John Levesque
Cray's Supercomputing Center of Excellence



The Target



ORNL's "Titan" System

- Upgrade of Jaguar from Cray XT5 to XK6
- Cray Linux Environment operating system
- Gemini interconnect
 - 3-D Torus
 - Globally addressable memory
 - Advanced synchronization features
- AMD Opteron 6274 processors (Interlagos)
- New accelerated node design using NVIDIA multi-core accelerators
 - 2011: 960 NVIDIA x2090 "Fermi" GPUs
 - 2012: 14,592 NVIDIA K20 "Kepler" GPUs
- 20+ PFlops peak system performance
- 600 TB DDR3 mem. + 88 TB GDDR5 mem

Titan Specs	
Compute Nodes	18,688
Login & I/O Nodes	512
Memory per node	32 GB + 6 GB
# of Fermi chips (2012)	960
# of NVIDIA K20 "Kepler" processor (2013)	14,592
Total System Memory	688 TB
Total System Peak Performance	20+ Petaflops
Cross Section Bandwidths	X=14.4 TB/s Y=11.3 TB/s Z=24.0 TB/s

The Challenge

Not the first Six

INCITE Awardees



How Effective are GPUs on Scalable Applications?

OLCF-3 Early Science Codes -- Current performance measurements on TitanDev



	XK6 (w/ GPU) vs. XK6 (w/o GPU)	XK6 (w/ GPU) vs. XE6	Cray XK6: Fermi GPU plus Interlagos CPU Cray XE6: Dual Interlagos and no GPU
Application	Performance Ratio	Performance Ratio	Comment
S3D	1.5	1.4(now 1.9)	<ul style="list-style-type: none"> Turbulent combustion 6% of Jaguar workload
Denovo	3.5	3.3	<ul style="list-style-type: none"> 3D neutron transport for nuclear reactors 2% of Jaguar workload
LAMMPS	6.5	3.2	<ul style="list-style-type: none"> High-performance molecular dynamics 1% of Jaguar workload
WL-LSMS	3.1	1.6	<ul style="list-style-type: none"> Statistical mechanics of magnetic materials 2% of Jaguar workload 2009 Gordon Bell Winner
CAM-SE	2.6	1.5	<ul style="list-style-type: none"> Community atmosphere model 1% of Jaguar workload

These are all Fermi+ numbers – Cannot show Kepler

The Approach



The Approach

- **Formulate a target problem that does real science and is same work/node as all MPI**

S3D – Weak Scaling Study

- **All MPI mesh on a node**
 - $15*15*15*32 = 108000$
- **One MPI tasks/node**
 - $48*48*48 = 110592$
- **Two MPI tasks/node**
 - $38*38*38*2=109744$

Share GPU with the two MPI tasks

The Approach

- Formulate a target problem that does real science and is same work/node as all MPI
- **Identify hotspots**
 - Using Craypat `-hprofile_generate` to identify loop structure

Re-compiling with `-hprofile_generate "pat_report-O callers"`

```

100.0% | 117.646170 | 13549032.0 |Total
|-----|
| 75.4% | 88.723495 | 13542013.0 |USER
|-----|
| 10.7% | 12.589734 | 2592000.0 |parabola_
|-----|
3|| 7.1% | 8.360290 | 1728000.0 |remap_.LOOPS
4|| | | | remap_
5|| | | | ppmlr_
|-----|
6|||| 3.2% | 3.708452 | 768000.0 |sweepx2_.LOOP.2.li.35
7|||| | | | sweepx2_.LOOP.1.li.34
8|||| | | | sweepx2_.LOOPS
9|||| | | | sweepx2_
10|||| | | | vhone_
6|||| 3.1% | 3.663423 | 768000.0 |sweepx1_.LOOP.2.li.35
7|||| | | | sweepx1_.LOOP.1.li.34
8|||| | | | sweepx1_.LOOPS
9|||| | | | sweepx1_
10|||| | | | vhone_
|=====|
3|| 3.6% | 4.229443 | 864000.0 |ppmlr_
|-----|
4||| 1.6% | 1.880874 | 384000.0 |sweepx2_.LOOP.2.li.35
5||| | | | sweepx2_.LOOP.1.li.34
6||| | | | sweepx2_.LOOPS
7||| | | | sweepx2_
8||| | | | vhone_
4||| 1.6% | 1.852820 | 384000.0 |sweepx1_.LOOP.2.li.35
5||| | | | sweepx1_.LOOP.1.li.34
6||| | | | sweepx1_.LOOPS
7||| | | | sweepx1_
8||| | | | vhone_
|=====|

```

The Approach

- Formulate a target problem that does real science and is same work/node as all MPI
- Identify hotspots
 - Using Craypat `-hprofile_generate` to identify loop structure
- **Convert all- MPI into Hybrid, using high level OpenMP**
 - Tune to beat the all MPI
 - Structure to be general purpose, n MPI tasks/node, m threads/node
 - Use OpenMP – easy path to the next step
 - **This is important for all future multi-petaflop systems**

Converting the MPI application to a Hybrid OpenMP/MPI application

Parallel Analysis, Scoping and Vectorization

- Current scoping tool, `-homp_analyze`, is meant to interface to a code restructuring GUI called “reveal”.
 - `!dir$ omp_analyze_loop`
- In order to utilize scoping tool for loops that contain procedures the program library need to be employed
 - `-hwp -hpl=vhone.aid`
 - This will do an initial pass of the code, checking for error and then at the load it will build the program library and perform the analysis
- Compiler will be very conservative

Main window of reveal

The screenshot shows the main window of the Reveal 0.1 software. The window title is "Reveal 0.1". The interface includes a menu bar with "File" and "Help". Below the menu bar, there are tabs for "About Reveal" and "vhone.aid". A "Full List" dropdown menu is visible, along with a search icon and a refresh icon. On the left side, there is a file browser showing a tree view of files: "riemann.f90", "states.f90", "sweepx1.f90", "sweepx2.f90", "sweepy.f90", "sweepz.f90", and a sub-directory "SWEEPZ" containing "Loop@22", "Loop@23", "Loop@24", "Loop@26", "Loop@54", "Loop@55", "Loop@62", "Loop@63", and "Loop@74". The "Loop@54" file is selected and highlighted in blue. The main area of the window displays the source code for "sweepz.f90". The code is as follows:

```

53 #endif
54 do j = 1, js
55   do i = 1, isz
56     radius = zxc(i+mypez*isz)
57     theta = zyc(j+mypey*js)
58     stheta = sin(theta)
59     radius = radius * stheta
60
61     ! Put state variables into 1D arrays, padding with 6 ghost zones
62     do m = 1, npez
63       do k = 1, ks
64         n = k + ks*(m-1) + 6
65         r(n) = recv3(1,j,k,i,m)
66         p(n) = recv3(2,j,k,i,m)
67         u(n) = recv3(5,j,k,i,m)
68         v(n) = recv3(3,j,k,i,m)
69         w(n) = recv3(4,j,k,i,m)
70         f(n) = recv3(6,j,k,i,m)
71       enddo
72     enddo
73
74     do k = 1, kmax
75       n = k + 6
76       xa(n) = zza(k)
77       dx(n) = zdz(k)
78       xa0(n) = zza(k)

```

At the bottom of the window, a status bar indicates "vhone.aid loaded".

Scoping window

OpenMP Construct

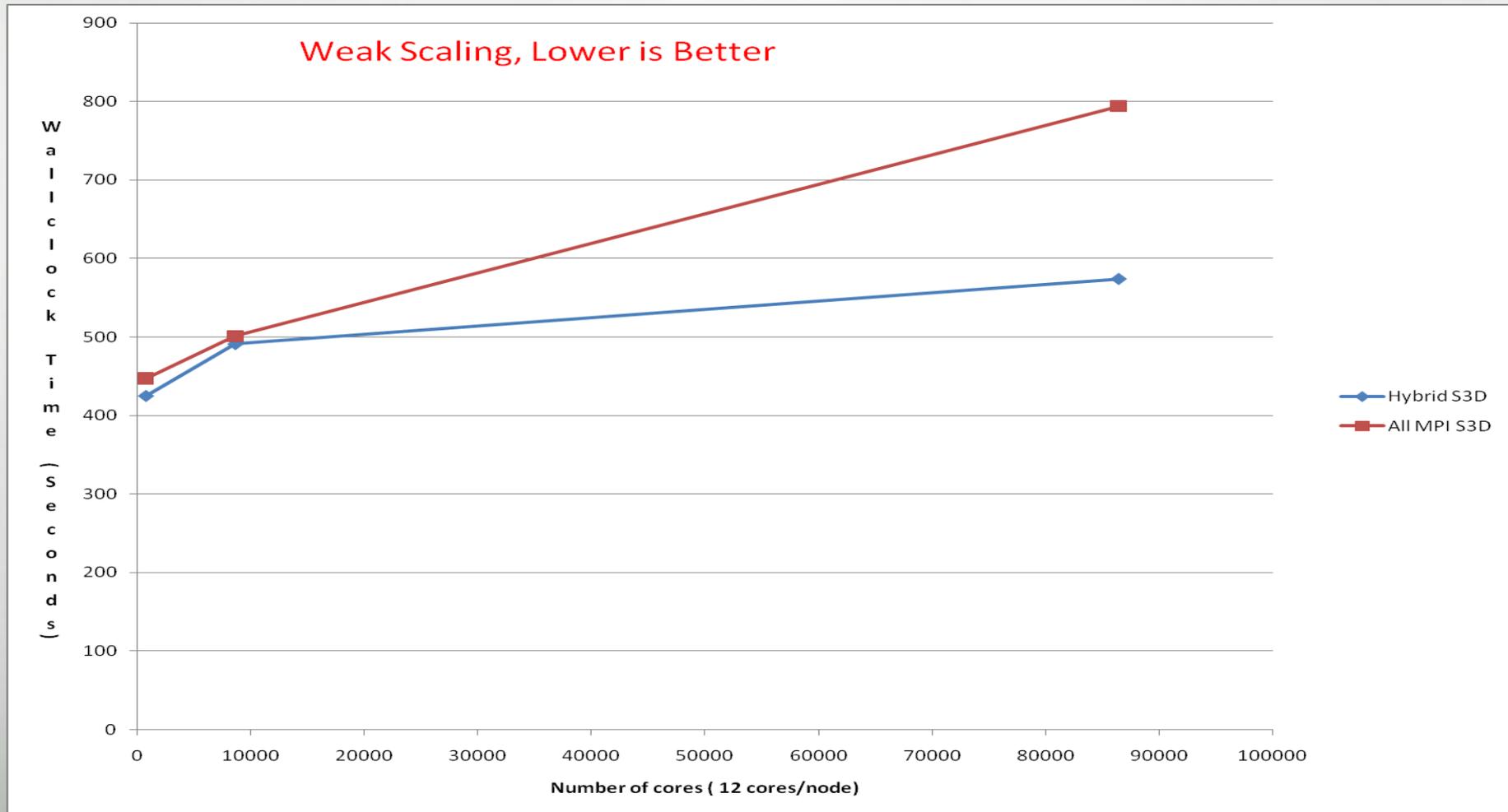
Name	Type	Scope	Info
zyc	Scalar	Unknown	
a	Scalar	Private	
ai	Scalar	Private	
amid	Scalar	Private	
ar	Scalar	Private	
b	Scalar	Private	
bi	Scalar	Private	
c	Scalar	Private	
cdtdx	Scalar	Private	
ci	Scalar	Private	
clft	Scalar	Private	
crgh	Scalar	Private	

Enable First Private
 Enable Last Private

Reduction:

Search:

Resultant Hybrid S3D Performance



The Approach

- Formulate a target problem that does real science and is same work/node as all MPI
- Identify hotspots
 - Using Craypat `-hprofile_generate` to identify loop structure
- **Convert all- MPI into Hybrid, using high level OpenMP**
 - This is important for all future multi-petaflop systems
- **Move to accelerator using OpenACC**
 - Insert OpenACC on major OpenMP loops
 - Examine Compiler feedback on data movement

Moving from OpenMP to OpenACC

- Things that are different between OpenMP and OpenACC
 - Cannot have CRITICAL REGION down callchain
 - Cannot have THREADPRIVATE
 - Vectorization is much more important
 - Cache/Memory Optimization much more important
 - No EQUIVALENCE
- Currently both OpenMP and OpenACC must be included in the source

```

#ifdef GPU
!$acc parallel loop private( k,j,i,n,r, p, e, q, u, v, w, svel0,&
!$acc&    xa, xa0, dx, dx0, dvol, f, flat, para,radius, theta, stheta)&
!$acc&    reduction(max:svel)
#else
!$omp parallel do private( k,j,i,n,r, p, e, q, u, v, w, svel0,&
!$omp&    xa, xa0, dx, dx0, dvol, f, flat, para,radius, theta, stheta)&
!$omp&    reduction(max:svel)
#endif

```

Compiler list for SWEEPX1

```

45.          #ifdef GPU
46.  G-----< !$acc parallel loop private( k,j,i,n,r, p, e, q, u, v, w, svel0,&
47.  G          !$acc&      xa, xa0, dx, dx0, dvol, f, flat, para,radius, theta, stheta)&
48.  G          !$acc&      reduction(max:svel)
49.  G          #else
50.  G          !$omp parallel do private( k,j,i,n,r, p, e, q, u, v, w, svel0,&
51.  G          !$omp&      xa, xa0, dx, dx0, dvol, f, flat, para,radius, theta, stheta)&
52.  G          !$omp&      reduction(max:svel)
53.  G          #endif
55.  G g-----< do k = 1, ks
56.  G g 3-----< do j = 1, js
57.  G g 3          theta=0.0
58.  G g 3          stheta=0.0
59.  G g 3          radius=0.0
62.  G g 3 g-----< do i = 1,imax
63.  G g 3 g          n = i + 6
64.  G g 3 g          r (n) = zro(i,j,k)
65.  G g 3 g          p (n) = zpr(i,j,k)
66.  G g 3 g          u (n) = zux(i,j,k)
67.  G g 3 g          v (n) = zuy(i,j,k)
68.  G g 3 g          w (n) = zuz(i,j,k)
69.  G g 3 g          f (n) = zfl(i,j,k)
71.  G g 3 g          xa0(n) = zxa(i)
72.  G g 3 g          dx0(n) = zdx(i)
73.  G g 3 g          xa (n) = zxa(i)
74.  G g 3 g          dx (n) = zdx(i)
75.  G g 3 g          p (n) = max(smallp,p(n))
76.  G g 3 g          e (n) = p(n)/(r(n)*gamm)+0.5*(u(n)**2+v(n)**2+w(n)**2)
77.  G g 3 g-----> enddo
79.  G g 3          ! Do 1D hydro update using PPMLR
80.  G g 3 gr2 I--> call ppmlr (svel0, sweep, nmin, nmax, ngeom, nleft, nright,r, p, e, q, u, v, w, &
81.  G g 3          xa, xa0, dx, dx0, dvol, f, flat, para,radius, theta, stheta)
82.  G g 3

```

Compiler list for SWEEPX1

ftn-6405 ftn: ACCEL File = sweepx1.f90, Line = 46

A region starting at line 46 and ending at line 104 was placed on the accelerator.

ftn-6418 ftn: ACCEL File = sweepx1.f90, Line = 46

If not already present: allocate memory and copy whole array "zro" to accelerator, free at line 104 (acc_copyin).

ftn-6418 ftn: ACCEL File = sweepx1.f90, Line = 46

If not already present: allocate memory and copy whole array "zpr" to accelerator, free at line 104 (acc_copyin).

ftn-6418 ftn: ACCEL File = sweepx1.f90, Line = 46

If not already present: allocate memory and copy whole array "zux" to accelerator, free at line 104 (acc_copyin).

ftn-6418 ftn: ACCEL File = sweepx1.f90, Line = 46

If not already present: allocate memory and copy whole array "zuy" to accelerator, free at line 104 (acc_copyin).

ftn-6418 ftn: ACCEL File = sweepx1.f90, Line = 46

If not already present: allocate memory and copy whole array "zuz" to accelerator, free at line 104 (acc_copyin).

ftn-6418 ftn: ACCEL File = sweepx1.f90, Line = 46

If not already present: allocate memory and copy whole array "zfl" to accelerator, free at line 104 (acc_copyin).

ftn-6416 ftn: ACCEL File = sweepx1.f90, Line = 46

If not already present: allocate memory and copy whole array "send1" to accelerator, copy back at line 104 (acc_copy).

The Approach

- **Formulate a target problem that does real science and is same work/node as all MPI**
- **Identify hotspots**
 - Using Craypat `-hprofile_generate` to identify loop structure
- **Convert all- MPI into Hybrid, using high level OpenMP**
 - This is important for all future multi-petaflop systems
- **Move to accelerator using OpenACC**
 - Insert OpenACC on major OpenMP loops
 - Examine Compiler feedback on data movement
- **Introduce data regions outside the timestep loop**
 - Now have to find all the code that accesses the arrays you want to reside on the accelerator
 - Looking at the PIN tool to help with that complex analysis. Especially when using derived types and C++ dynamic arrays

Profile of Accelerated Version



Table 1: Time and Bytes Transferred for Accelerator Regions

Acc Time%	Acc Time	Host Time	Acc Copy In (MBytes)	Acc Copy Out (MBytes)	Calls	Function
						PE=HIDE Thread=HIDE
100.0%	58.363	67.688	24006.022	16514.196	14007	Total

30.3%	17.697	0.022	--	--	1000	sweepy_.ACC_KERNEL@li.47
22.0%	12.827	0.010	--	--	500	sweepx2_.ACC_KERNEL@li.46
21.2%	12.374	0.013	--	--	500	sweepz_.ACC_KERNEL@li.67
14.0%	8.170	0.013	--	--	500	sweepx1_.ACC_KERNEL@li.46
3.9%	2.281	1.161	12000.004	--	1000	sweepy_.ACC_COPY@li.47
2.0%	1.162	0.601	6000.002	--	500	sweepz_.ACC_COPY@li.67
1.6%	0.953	0.014	--	6000.004	1000	sweepy_.ACC_COPY@li.129
1.0%	0.593	0.546	3000.002	--	500	sweepx1_.ACC_COPY@li.46
1.0%	0.591	0.533	3000.002	--	500	sweepx2_.ACC_COPY@li.46
0.8%	0.494	0.015	--	3000.002	500	sweepx2_.ACC_COPY@li.107
0.8%	0.485	0.007	--	3000.002	500	sweepx1_.ACC_COPY@li.104
0.8%	0.477	0.007	--	3000.002	500	sweepz_.ACC_COPY@li.150
0.4%	0.250	0.016	--	1503.174	500	vhone_.ACC_COPY@li.251
0.0%	0.005	0.005	6.012	--	1	vhone_.ACC_COPY@li.205
0.0%	0.001	0.000	--	6.012	1	vhone_.ACC_COPY@li.283
0.0%	0.001	0.000	--	5.000	1	vhone_.ACC_COPY@li.266
=====						

The Approach

- **Formulate a target problem that does real science and is same work/node as all MPI**
- **Identify hotspots**
 - Using Craypat `-hprofile_generate` to identify loop structure
- **Convert all- MPI into Hybrid, using high level OpenMP**
 - This is important for all future multi-petaflop systems
- **Move to accelerator using OpenACC**
 - Insert OpenACC on major OpenMP loops
 - Examine Compiler feedback on data movement
- **Introduce data regions outside the timestep loop**
 - Now have to find all the code that accesses the arrays you want to reside on the accelerator
 - Looking at the PIN tool to help with that complex analysis. Especially when using derived types and C++ dynamic arrays
- **Optimize the kernels**

- A common directive programming model for **today's GPUs**
 - Announced at SC11 conference
 - Offers portability between compilers
 - Drawn up by: NVIDIA, Cray, PGI, CAPS
 - Multiple compilers offer portability, debugging, permanent
 - Works for Fortran, C, C++
 - Standard available at www.OpenACC-standard.org
 - Initially implementations targeted at NVIDIA GPUs
- Current version: 1.0 (November 2011)
- Compiler support:
 - Cray CCE
 - PGI Accelerator
 - CAPS



Creating arrays on Accelerator using runtime calls

```

#ifdef _OPENACC

fdsim_ac3dvti *sim_fwd;
sim_fwd = (fdsim_ac3dvti*)(pd.sim_fwd);
void *data;
int size;
for ( int isub=0; isub < pd.sim_fwd->nsubdom; isub++ ) {
  if(sim_fwd->pml_coef[isub] != 0){
    for(int i=0 ; i<12 ; i++){
      if(i>=0 && i<=3)size = sim_fwd->txx[isub]->ax1->ntot;
      if(i>=4 && i<=7)size = sim_fwd->txx[isub]->ax2->ntot;
      if(i>=8 && i<=11)size = sim_fwd->txx[isub]->ax3->ntot;
      data = sim_fwd->pml_coef[isub][i];
      cray_acc_create(data,size*sizeof(float));
    }
  }
}

```

Updating arrays on Accelerator using runtime calls

```

#ifdef _OPENACCX
#include "openacc.h"
void fdsim_ac3dvti::acc_updatein_arrays(){
    acc_update_arrays(1);
}

void fdsim_ac3dvti::acc_updateout_arrays(){
    acc_update_arrays(2);
}

void fdsim_ac3dvti::acc_update_anarray(int inorout, void* data, size_t size){
    if(inorout==1){
        cray_acc_copyin(data,size);
    }else{
        cray_acc_copyout(data,size);
    }
}
}

```

What does OpenACC look like

```

! X - direction communication - (+) side
reqcount = 0
do i = 1, derivcount
  if( deriv_x_list(i)%pos_nbr >= 0 .and. &
    deriv_x_list(i)%inuse ) then

    reqcount = reqcount + 1
    req(reqcount) = deriv_x_list(i)%req(3)

  endif
enddo
if( reqcount > 0 ) then
  !write(*,'(1i4,1a,1i4)') myid, 'x pos waiting on ', reqcount
  call MPI_WAITALL( reqcount, req, stat, ierr )
#endifdef GPU_ASYNC
!$acc update device(pos_f_x_buf(:, :, :, 1:reqcount)) async(1)
#endifif
endif

```

Using GPU Direct

```

if(lnbr(1)>=0) then
  ! get ghost cells from neighbor on (-x) side
#ifdef GPU_DIRECT
!$acc host_data use_device(neg_f_x_buf)
#endif
      call MPI_IRecv(neg_f_x_buf(1,1,1,idx), (my*mz*iorder/2), &
                    MPI_REAL8, deriv_x_list(idx)%neg_nbr, idx, &
                    gcomm, deriv_x_list(idx)%req(1), ierr)
#ifdef GPU_DIRECT
!$acc end_host_data
#endif
endif
  if(lnbr(2)>=0) then
    ! get ghost cells from neighbor on (+x) side
#ifdef GPU_DIRECT
!$acc host_data use_device(pos_f_x_buf)
#endif
      call MPI_IRecv(pos_f_x_buf(1,1,1,idx), (my*mz*iorder/2), &
                    MPI_REAL8, deriv_x_list(idx)%pos_nbr, idx+deriv_list_size, &
                    gcomm, deriv_x_list(idx)%req(3), ierr)
#ifdef GPU_DIRECT
!$acc end_host_data
#endif
endif
endif

```

Do not have to update host with data prior to communication

Using GPU Direct

```

if( deriv_x_list(idx)%packed ) then
    ! assume pos_fx_x_buffer and neg_fs_x_buffer have been filled somewhere
    ! else
    if(deriv_x_list(idx)%neg_nbr>=0) then
        ! send ghost cells to neighbor on (-x) side
#ifdef GPU_DIRECTS
!$acc host_data use_device(pos_fs_x_buf)
#endif
        call MPI_ISend(pos_fs_x_buf(1,1,1,idx), (my*mz*iorder/2), &
                     MPI_REAL8,deriv_x_list(idx)%neg_nbr,idx+deriv_list_size,&
                     gcomm,deriv_x_list(idx)%req(2),ierr)

#ifdef GPU_DIRECTS
!$acc end host_data
#endif
    endif
    if(deriv_x_list(idx)%pos_nbr>=0) then
        ! send ghost cells to neighbor on (+x) side
        nm = mx + 1 - iorder/2
#ifdef GPU_DIRECTS
!$acc host_data use_device(pos_fs_x_buf)
#endif
        call MPI_ISend(neg_fs_x_buf(1,1,1,idx), (my*mz*iorder/2), &
                     MPI_REAL8,deriv_x_list(idx)%pos_nbr,idx, &
                     gcomm,deriv_x_list(idx)%req(4),ierr)

#ifdef GPU_DIRECTS
!$acc end host_data
#endif
    endif
endif

```

Do not have to update host with data prior to communication

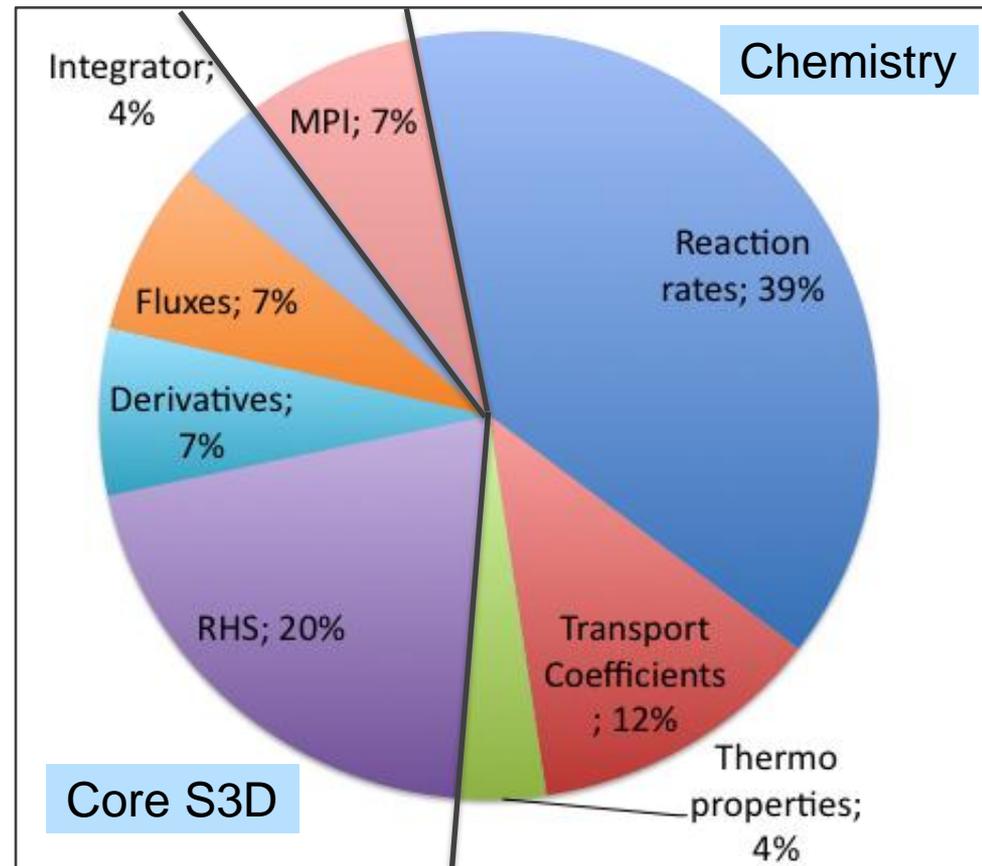
And the principal good thing about S3D

A benchmark problem was defined to closely resemble the target simulation

- 52 species n-heptane chemistry and 48^3 grid points per node

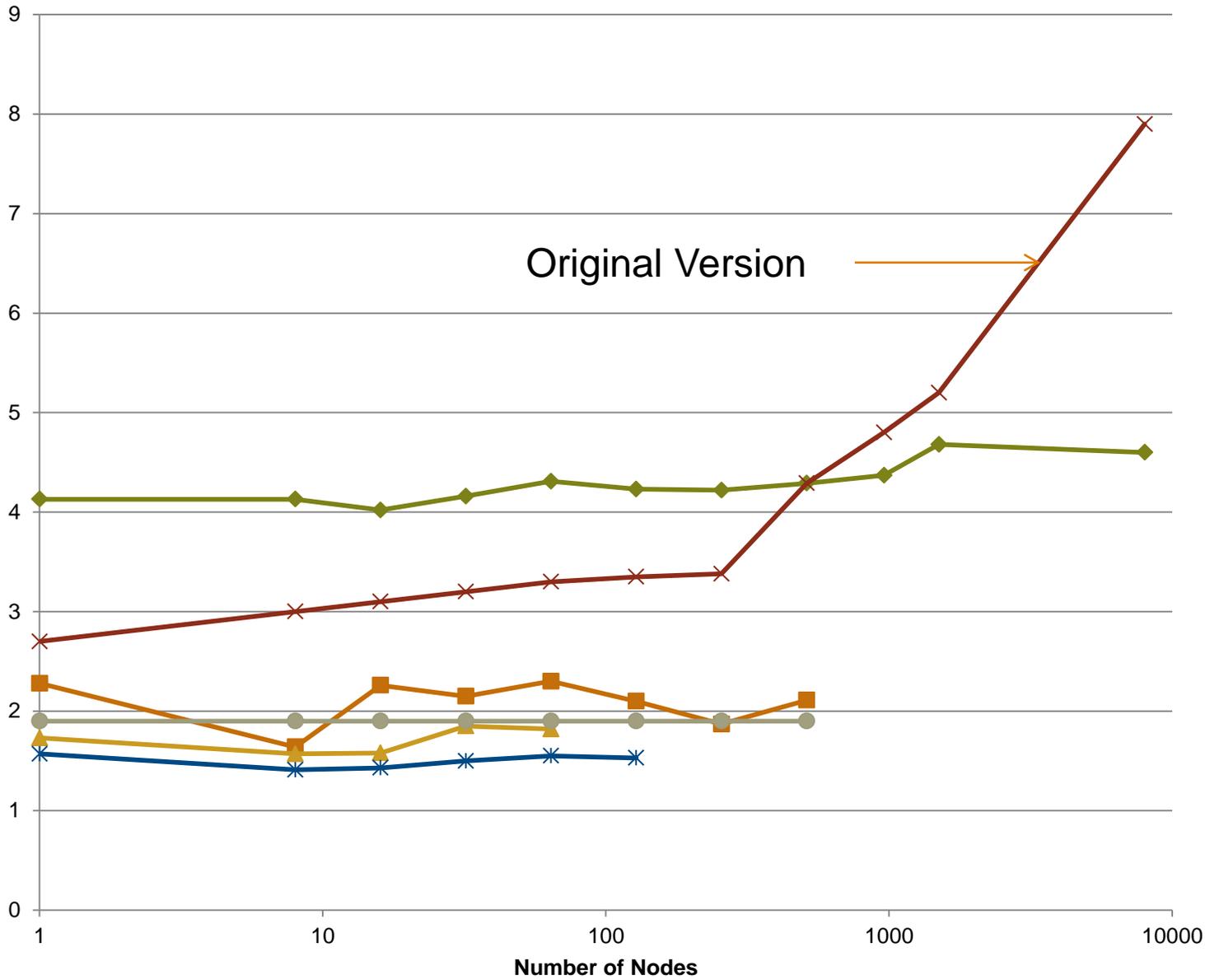
- $48^3 * 18,500$ nodes = 2 billion grid points
- Target problem would take two months on today's Jaguar

- Code was benchmarked and profiled on dual-hex core XT5
- Several kernels identified and extracted into stand-alone driver programs





Comparisons of Various systems running S3D



Acc Time%	Acc Time	Host Time	Acc Copy In (MBytes)	Acc Copy Out (MBytes)	Events	Function
100.0%	105.999	172.083	55959	100021	1472704	Total
21.2%	22.503	0.077	--	--	600	reaction_rate_vec_.ACC_KERNEL@li.167
3.7%	3.891	0.035	--	--	600	rhsf_.ACC_KERNEL@li.1756
3.5%	3.672	0.020	--	--	600	rhsf_.ACC_KERNEL@li.1820
3.3%	3.451	0.021	--	--	600	rhsf_.ACC_KERNEL@li.1875
2.9%	3.083	0.009	--	19491	100	integrate_.ACC_COPY@li.74
2.8%	3.005	10.686	--	6497	92400	derivative_z_pack_np_.ACC_COPY@li.573
2.8%	2.996	10.636	--	6497	92400	derivative_z_pack_np_.ACC_COPY@li.623
2.6%	2.783	0.068	--	--	600	rhsf_.ACC_KERNEL@li.439
2.5%	2.605	0.019	--	--	600	rhsf_.ACC_KERNEL@li.654
2.4%	2.559	0.004	--	--	100	computecoefficients_r_.ACC_KERNEL@li.148
2.3%	2.444	0.035	13162	--	600	rhsf_.ACC_COPY@li.417
2.3%	2.431	0.041	12909	--	600	rhsf_.ACC_COPY@li.1870
2.3%	2.419	0.039	12909	--	600	rhsf_.ACC_COPY@li.1871
2.3%	2.404	5.235	--	7341	600	save_bc_deriv1\$rhsf_.ACC_COPY@li.248
2.1%	2.254	2.138	0.721	--	600	rhsf_.ACC_COPY@li.256
1.9%	2.028	0.029	--	--	700	calc_primary_vars_.ACC_KERNEL@li.42
1.8%	1.949	0.004	--	--	100	computecoefficients_r_.ACC_KERNEL@li.234
1.7%	1.766	10.701	--	--	92400	derivative_z_pack_np_.ACC_KERNEL@li.557
1.7%	1.760	0.014	--	--	600	rhsf_.ACC_KERNEL@li.992
1.6%	1.727	0.245	--	6497	1800	derivative_x_pack_np_.ACC_COPY@li.658
1.6%	1.690	10.646	--	--	92400	derivative_z_pack_np_.ACC_KERNEL@li.607
1.4%	1.434	0.235	--	6497	1800	derivative_y_pack_np_.ACC_COPY@li.624
1.3%	1.387	0.078	--	--	600	rhsf_.ACC_KERNEL@li.488
1.3%	1.328	1.433	2953	--	700	calc_primary_vars_.ACC_COPY@li.42
1.2%	1.247	0.223	--	6497	1800	derivative_x_pack_np_.ACC_COPY@li.613
1.1%	1.137	0.019	--	--	600	integrate_.ACC_KERNEL@li.123
1.0%	1.069	0.245	--	6497	1800	derivative_y_pack_np_.ACC_COPY@li.672

And that is not all

Titan will be delivered with Nvidia Kepler, which will give us better performance

Clock Rate

Memory Bandwidth

More Registers

GPU direct will be available from MPI with final delivery

This will give us a very good increase in performance

Cuda Proxy allows for multiple MPI tasks to share GPU on the XK system

Cray Technical Workshop on XK6 Programming

On October 9-10, 2012, ORNL and Cray will be hosting a two day XK6 programming workshop. The intent is to bring together users of XK6 systems around the world and share experiences. We would very much like your participation in this workshop. In addition to user talks we will have expert panels covering programming models for the XK6 including OpenACC, Cuda, OpenCL, Cuda Fortran, etc.

Attendance will be limited to 75 people, so please sign up early. The workshop will be held in the JICS auditorium on the ORNL campus and foreign nationals will need to submit visitor requests six weeks prior to the event. Please consider attending and if you know of others who would be interested in either attending or speaking, please forward this email.

Registration is required for this event. Information about travel and accommodations in the Oak Ridge area are provided below. Refreshments (not lunch) will be provided during the event. Additionally, the event will be webcast. Please indicate if you intend to participate in person or via webcast on the registration form.